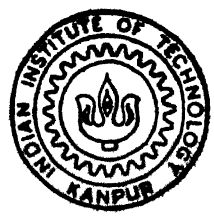
PATH PLANNING OVER NATURAL TERRAIN USING DIGITAL TERRAIN MODEL

By

Capt. A. A. BHAT



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

FEBRUARY, 1993

PATH PLANNING OVER NATURAL TERRAIN USING DIGITAL TERRAIN MODEL

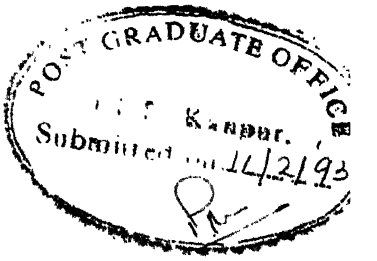
A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

By
Capt A. A. BHAT

to the
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
February, 1993

CSE-1993-M-BHA-PAT



Certificate

It is certified that the work contained in the thesis entitled *Path Planning over Natural Terrain Using Digital Terrain Model*, by *Capt A. A. Bhat*, has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

Dr. R. M. K. Sinha,
Professor,
Department of Computer
Science and Engineering
I.I.T. Kanpur - 208016.

Dr. Sanjay G. Dhande,
Professor,
Department of Computer
Science and Engineering,
I.I.T. Kanpur - 208016.

February, 1993.

Acknowledgement

I would like to express my deep sense of gratitude to my guides, Dr. R. M. K. Sinha and Dr. Sanjay G. Dhande for their expert guidance and constant encouragement throughout this thesis work. The brainstorming sessions I had with them not only developed my basics in Artificial Intelligance and Computer graphics and modeling, but also helped me evolve new ideas and smoothen out the rough edges.

I am also grateful to the lab staff of CSE deprtment and the software engineers at the computer center for their assistance and cooperation during this work.

I would also like to thank Mr. Sen and others at the Robotics lab for providing me the access to the literature held in their lab.

I would also like to mention the help provided by Dr. Chansarkar, Mr. Panchal of the D. T. R. L in formulating the problem and giving a user end perception of the problem.

Special thanks to Capt K. T. Sreekumar, whose timely suggestions and support enabled a smooth passage throughout the M.tech programme.

Thanks also to Mr. Sudheer Deshpande, Mr. P. S. Avadhani, Mr. Harshad Parikh and my other collegues of M.Tech(CS) 92-93 batch, who sometime or the other "bailed" me out of some irritating "bug", during the development of the software.

Behind every successful man, there stands a woman (who keeps telling him ... "he's wrong"!). I could not have been able to complete this work without the support of my wife, Arundhati, who, despite her own domestic troubles, kept encouraging me and telling me..."I was on the right track".

Capt A. A. Bhat

Contents

List of Figures

List of Tables

Abstract

Path Planning of a mobile robot has been a topic of research for many scientists, involved with development of mobile robots or autonomous vehicles. The main thrust of most of the research has been obstacle avoidance, or movement through discrete, homogeneous, weighted regions.

When a leg-powered human, a petrol driven car or a diesel truck has to travel over a surface consisting of hills and valleys, dry deserts and thick forests; the work done (cost of traversal) per unit distance varies for every yard of progress made. The regions are no more homogeneous and this variable *gradient* cost has to be included in the overall traversal cost.

Movement of military convoys and troops to remote areas, construction of roads in such areas etc. are some areas where the present work can be applied/used.

This report proposes a method for *path planning* over a *natural terrain*, accounting for the costs due to *variable gradients*. Heuristics for encountering exceptionally high cost, or forbidden regions are also integrated with this method. This is achieved by sectionalizing the search, at the vertices of the forbidden regions.

A digital terrain model, which not only suites the requirements of the *path planning* method, but also effects saving in storage space, is used here.

The heuristics applied here are based on experimental learning and are expected to give polynomial time, near-optimal solutions.

Chapter 1

Introduction

1.1 Path Planning—an overview

In the realm of AI and Robotics, the two emerging fields of today, lies an interesting area—*path planning*, which promises to explore, unfold and simulate the mysteries involved with the intriguing aspects of *planning* in the human brain. The expertise of a human brain to analyze incomplete and imprecise information and use this information for *path planning* is in itself remarkable; it is difficult to visualize a machine gaining a part of this expertise.

Many problems of engineering and scientific importance can be related to the general problem of " finding a *path* through a graph , from a START node to a GOAL node". Routing of telecommunications traffic through the telecom network, layout of printed circuit boards, routing in computer networks, mechanical theorem proving and problem solving are some examples of *path planning*. *Robotics* is the area which pioneered the interest in *path planning*, has the biggest and most varied application of *path planning* [Charniac 85]. Some of these problems in *Robotics* are :-

- Motion of a mobile robot in a cluttered indoor planar workspace involving collision avoidance/obstacle avoidance strategies.
- Movement of a robot in a multistation workspace using an optimal schedule, adhering to deadlines and time constraints.
- Motion/trajectory of robot links in 3D free space.
- Motion of robot on rough terrains, involving weighted cost areas.
- Move of Autonomous Land Vehicle on different non-planar surfaces.

- Motion of legged robots/walking machine on a rough terrain—interesting dimension added by the capability of a walking machine to hop/cross over minor obstacles.

Applications of methods similar to *path planning* have also generated great interest in some areas are discussed below.

1.1.1 Traffic planning

In many big cities and over-populated areas there exists a maze of roads (which are like edges in a graph). The city planners have to plan the traffic flow using *no-entries* and *one-ways*. Problem of planning traffic flow during special occasions like an international sports event, national day parades, rock show etc. can cause nightmares to any planner.

1.1.2 Computer and Communication Networks

A routing problem in a computer network is a classical *path planning* problem . The basic constraint here is the bandwidth capacity of a link; congestion avoidance, reliable delivery of messages/packets and optimal speed of transfer being the criterion for optimality of a path.

1.1.3 Game Playing

Path Planning as applied to game playing involves generating procedures so that valid moves (or paths) are recognized and a best move is explored first[Rich 83]. The selection heuristics has to account for an opponent, (worst case is a human opponent) who is equally smart.

1.1.4 Roads/Tracks/Rail Routes Planning and Construction

Roads,tracks, rails, passes etc. require a high level of cartographic planning. A route planned optimally using *path planning* techniques can not only save on time, material cost of construction and hundreds of pairs of jungle boots used up while reconnoitering, but can also reduce the hazards faced by the workers.

1.2 Path Planning over Natural Terrain

Most *path planning* problems mentioned earlier have been attacked by splitting the problem in two subproblems [Rowe 90, Hart 68].

1. Reducing the problem space into a finite graph.

2. Applying a suitable search strategy to find an optimal or near optimal solution[1].

First of the subproblem becomes harder as the working environment becomes more complex. In recent years, most research is oriented in this direction. The second subproblem is satisfactorily solved by applying standard search methods like the A* and Dijkstra's uninformed 'Bestfirst' method.

Path Planning on a natural terrain is quite complex in terms of reduction to a finite graph. An ungainly graph size would lead to inefficient, wasteful search; while adding more complexity to the subproblem 1 and limiting the graph size may lead to suboptimal results.

1.2.1 Problem Statement

The problem of *path planning* on a *natural terrain* can be broken up further and stated as follows:-

1. Suitable three dimensional representation [2] of the natural terrain.
2. Generating a *finite* graph from a START to a GOAL.
3. Searching for the optimal solution.
4. Fitting the solution appropriately on the 3-D representation.

1.2.2 Issues Involved

The *path planning* on a *natural terrain* can be looked at as a generalization of *path planning* over different surfaces. However there are some peculiar issues related with the problem, which increase its complexity.

The Terrain

The natural terrain in itself is a diverse problem space. Various features of the terrain are traditionally represented on a two dimensional map. These features are called *cartographic features*. There are some other features of importance which cannot be displayed on a single map or require overlaying of a special map on the conventional cartograph. These features are the *thematic features* of the terrain.

[1] In many applications of robotics, collision avoidance being more important, the first satisficing solution is considered good enough

[2] Terrain should be represented in 3-D and as non-homogenous. Most solutions in *Robotics* [Charniac 85] [Rowe 90] use a 2-D, homogeneous representation and featurification of natural terrain.

- *Cartographic Features.* The most important cartographic feature addressed here is the *Gradient.* The *gradient* though closely related to altitude[3] is considered instead of altitude, as for any mobile agent, the altitude it is traveling is unimportant as long as it travels on a *plateau.* The factor of reduction in speed is related to the *gradient* on which it is to travel. It is this feature which has not been given sufficient importance in the work on *path planning* . The *gradient* forces the algorithms to look at the third physical dimension, thus de-homogenizing the surface of propagation. Other features such as the roads, rivers, bridges, villages, built up areas etc. can be dealt with by two dimensional heuristic methods([Rowe 90]). The existence of such features have varied effect on the plausible path costs, depending on the type of mobile agent. A road feature along the direction of propagation may provide a low cost path for wheeled automobiles, while it may be inconsequential for a cross-country trekker.

- *Thematic Features.* Inclusion of most of the thematic features are agent[4] and application dependant. The thematic features are generally represented by 2D polygons of differing homogeneous cost. Some of them may be totally prohibitive for movement(e.g a reserved forest) while most will incur a higher cost per unit distance of traversal. Some example themes and their implications are listed below.

    1. *Soil Type.* The soil under the wheels of a 4 wheeler can affect the speed of its propagation. Additional cost for certain types of soil due to climatic conditions can also accrue.

    2. *Land use.* Land use polygons represent the surface cover of a piece of land. Certain types of land uses may be prohibited for trespassing. Land use polygons can also cause reduction in path cost in "building/construction" applications if these polygons turn into raw material (mainly wood) providers.

    3. *Geology.* Geological polygons give information on the rocks underneath the surface . This theme is important when *path planning* is to be used for "Road construction" purposes, and also has some bearing on movement of vehicles.

[3] Gradient is the rate of change of altitude at a given point in a given direction.
[4] *agent* is referred to as the mobile agent which has to negotiate the terrain.

Database

A "knowledge base" extracted from a variety of data available in different forms will have to be employed for efficient use in the *path planning* method. Preprocessing for standardization and quantization will have to be carried out, on the raw data, which has to be drawn from diverse sources. Types of data required by the knowledge base are listed as follows:-

- *Elevation Data.* The sheer size of the database required to store this data has prompted use of several space saving techniques. Storing the elevation data at regularly spaced longitudes and latitudes (regular grid method) is one of the most common methods. However by this method, intermediate, important features may be ill-represented, while tracks of gently sloped ground are represented wastefully by a dense grid.

  Irregular grids (DEMs[Guptill 79]) are employed to overcome this problem, but require painstaking methods to digitize.

  Storage methods in form of trees (specially quadtrees) have also been used for *path planning* applications[Mitchel 88], but require additional run time processing.

  Another method uses a coarse regular/irregular grids, and uses fractal methods to generate intermediate point elevations(The method is discussed in the next chapter).

- *Linear Map Information.* Linear features such as roads, rivers, hydrography and boundary regions, all can be stored as Digital Line Graphs (DLGs[Guptill 79]) or in forms of linked lists or indexed arrays.

- *Thematic Knowledge.* Existence of thematic polygons have different implications for different agents under differing climatic conditions. The information that exists in textual form in books, manuals and encyclopedias, has to be quantized into cost and weightage functions by an expert. Machine learning techniques also can be employed to devise methods of creating the database. This information has to be extracted and associated with the thematic polygons in a DLG.

1.3 Literature Review

1.3.1 Review of Terrain Modeling

In 1.2.1 above, modeling the terrain has been shown as the first subproblem. *Surface modeling* is a well researched field in Computer Graphics. *Natural Terrain* differs from other standard surfaces in that it does not have simple macroscopic characteristics as well as regular surface features[Fournier 82]. Conventionally, a Digital Terrain Model(DTM), was represented as a simple, regular grid overlayed over a 2-D map, with elevation data at each grid vertex, stored in a database or a 2-D array. Fowler and Little[Fowler 79] have contested the validity of such approach giving similar arguments as in 1.2.2. They have selected an *Irregular Triangulated Network* Model(TIN) to represent the *natural terrain* and have proposed a method for automatic[5] extraction of the grid points in the network. Marshall and Wilson[Marshall 80] have used a procedural model to combine fundamental data elements in creation of unified objects comprising the terrain model. A whole landscape is shown to be an aggregation of base element types such as a tree, a mountain etc. .

Fractal Models

Fractal models for the *natural terrain* were first developed by Mandelbrot, [Mandelbrot 82] highlighting the property of self-similarity found in nature. Fractal Brownian Motion in different dimensions has been used to model different natural phenomena. Fournier and Russell[Fournier 82] have analyzed the Mandelbrot concept, defining *natural terrain* in terms of a *stochastic process*. They have proposed the " Recursive Subdivision Algorithm " which has been used in most implementations later. Saupe [Saupe 88] has illustrated the use of *fractal Brownian motion* principal in three classes of algorithms, used for random fractal images.

1. Midpoint Displacement Algorithms which are similar to those proposed in [Fournier 82].
2. Fourier Filtering Methods.
3. Random Cut Methods.

Graphic issues for implementation of these methods have also been discussed in this article. Dong, Peng and Liang, [Dong 90] have claimed to provide smoother, more adaptable

[5] for an irregular grid a manual selection of grid vertices was commonly used

and more realistic model through a functional modeling approach called the *delta functional model.* The scene creation is achieved through stochastic modeling of natural processes, such as pressing and vibrating of the earth's crust and washing by rain.

1.3.2 Related Work in Robotics

Path planning has a major impact in *Robotics* (see 1.1). The simpler problem of, "Planning of collision free Path among polyhedral obstacles for a mobile robot", has been first addressed by Lozano-Peres et. al. [Lozano 79]. [6] A visibility graph (VGRAPH) is formed by connecting all the vertices of obstacles, *Visible* from each other and the START and The GOAL nodes. The path is obtained by running a standard search algorithm(like the A*) on this graph and then backtracking the solution path. The main problem tackled here is the non-trivial shape of the moving agent. A method of *growing* the obstacles in relation with the shape and likely directions of move of the agent[7] is proposed.

The above method does not give due consideration to the optimality criterion, which has been addressed to a certain extent(the local optimality part) by Khatib [Khatib 86] . A penalty function is associated, for presence of an obstacle in near vicinity. The obstacles are visualized as electromagnetic repellers with a potential field around them, which follows the *inverse square law.* The path generated follows a local optimality criteria and tends to keep away from obstacle edges and corners. Bagchi[Bagchi 91] has addressed the problem of *path planning* in dynamic environment and advocated application of a fuzzy logic method for collision avoidance in this environment.

Mitchel's research[Mitchel 88] has been one of the most wide ranged. He has addressed not only the collision avoidance problem, but also other associated problems such as *Weighted Region*, *concealment problem* etc. . Starting with the various methods of modeling the *natural terrain* he has discussed the complexity and optimality issues involved with the the various VGRAPH methods(see 1.3.2). He then discusses another obstacle avoidance method called the em Shortest Path Map Method, in which the problem space is recursively subdivided into optimal regions. This, he proves to have a better complexity($O(n \log n)$) as compared with the VGRAPH methods ($O(n^2 \log n)$). He has also discussed the methods for finding path on a non-planar, non-convex surface(earlier methods are extrapolated by discretizing the surface). Many methods have been proposed to solve the *weighted regio*

[6] Also see [Charniac 85].

[7] The agent then shrinks to a point size

problem. Some of these are *Weighted grid pixel method, Polygonal subdivision method* and *Snell's Law method* .

The Snell's Law method is also proposed by Rowe and Richbourg [Richbourg 90]. The optimal path is modeled as a light ray passing through different media. The analogy compares the traversal cost/per unit distance of the weighted regions, to the refractive index of light through different media. As the light rays travel through a medium in a straight line (shortest distance), and are subjected to refraction and internal reflection, the analogous optimal path travels through a single weighted region in a straight line and is subjected to refraction and total internal reflection. Rowe[Rowe 90] is also credited to the development of heuristics to deal with the linear features such as roads, rivers etc. . The Heuristics called *Shortcut Meta-heuristics* exploit the local optimality criteria, the path must follow. These criteria are applied at turns, crossings etc. of the rivers and roads, ensuring that the turns are within the *critical angle for total internal reflection.*

1.3.3 Searching and AI

Most of the work reviewed in 1.3.2 uses some state-space search approach, once the graph for the search is generated.

A problem can be defined as a state-space search problem, in a step–by–step manner. This is done by identifying the entities such as initial state, goal states, operators to be applied and rules governing the search(see [Rich 83], for detailed discussion). a state-space search conventionally follows the two fundamental techniques of *depth first search* or *breadth first search* or some modifications of the same (see [Rich 83],[Nilsson 85], for their discussion). Adding heuristics to aid the search process, has a considerable impact on the search-time, optimality and the graph size (see [Rich 83]).

The A* algorithm for the heuristic, *Best first search* has been proposed first by Hart et. al. [Hart 68]. This paper also includes a detailed discussion on the admissibility, optimality and complexity of the algorithm, which has been reproduced by Nilsson in [Nilsson 85]. Rich [Rich 83] gives the detailed statement of the algorithm. Discussion on the path cost function g' and heuristic function h' is found in [Nilsson 85], while the effects of over or under estimating h' are illustrated in [Rich 83]. Other heuristic search methods such as the *staged search* are explained in [Nilsson 85].

Heuristic power of a search technique is rather difficult to quantify, though comparison

of two techniques is possible. Nilsson [Nilsson 85] defines two parameters P and B for such comparison. P, the measure of penetrance of a search is defined as:-

$$P = L/T, P <= 1.$$

where L is the length of the optimal path and T the total numbers of nodes generated during the search. B is the effective branching factor is related to L and T by

$$B + B^2 + ... + B^L = T.$$

1.3.4 GIS

A *Geographical Information Systems*(GIS) contains the whole gambit of geography, terrain and geology related information. A *path planning* method has applications in GIS as a tool, to answer many of the queries to the system. Nature of some of the queries is found in [Phillips 79].

Various methods of designing and maintaining a GIS or a *geographical database* are discussed by [Juan 79], [Jimenez 79], [Mantey 79] and [Guptill 79].

One of the fast emerging applications of GIS is in the field of environmental mapping. Puttre [Puttre 92] discusses the various aspects of a GIS and illustrates it's impact on environmental mapping.

1.4 Objectives of the Present Work

One of the major problems a military commander in field, faces, is the transportation of equipment, supplies and personnel over unknown terrain. He has insufficient data, limited expertise and extremely short time to plan such moves. The only aids he has are the contour map and the prismatic compass. A computerized solution to his problem may relieve him of this burden, allowing him more time for important tactical decisions. The objectives for this work were inspired by this final aim in mind. The objectives are listed as below:-

- To develop a Digital Terrain Model, suitable for construction of path finding method as well as for realistic three dimensional rendering on the computer screen.
- Develop a suitable graph generation method for generating an optimal path, constrained by the following terrain features:-

1. Variable, non-discreete gradient.
2. High cost, weighted regions.
3. Prohibitive regions.
4. Multi-theme inputs to be aggregated for 2 and 3 above.

- Implement an efficient search method to evolve an optimal/near optimal path from a START point to the GOAL.

1.5 Scope and Organization

In this report the heuristics involved in the *path planning* method developed, have been proposed and explained, comparing it with the earlier work in this field. A method of modeling the *natural terrain* to suit the requirements of the *path planning* techniques has also been proposed. A software module, developed to test and try out the method has also been discussed in this report.

The report is organized into four chapters. First chapter deals with various aspects of depicting the natural terrain in the form of a *digital terrain model.* The chapter first discusses traditional models of modeling the terrain, and then explains the *fractal* method of modeling the terrain, which has been used in this work. The creation and overlaying of a *grid* on the terrain map is discussed subsequently, and to complete the discussions on modeling aspects, method extraction of relevant information, such as the elevation data is elaborated.

The second chapter is devoted to the statement and analysis of the *path planning* heuristics. In the first section of the chapter the problem is defined as a state-space search problem. The A* algorithm statement, in detail, is stated in the next section. Implementation details of the algorithm, along with graph generation heuristics are included next. The *staged search* is an additional modification, required while dealing with high cost and forbidden regions. This heuristic method of sectionalizing the graph and carrying out a piecemeal search is discussed in the next section. In the end of the chapter, the complexity and efficiency issues of the new technique, have been discussed.

Chapter 3 discusses the software development issues. The issues discussed here include the graphic issues in generating the DTM and display/rendering aspects of the optimal path, the database and datastructures used for the trial software developed during the course of

this work, and the interface with the user required for such systems. The chapter also has a mention of trials and tests carried out on some simple geometrical surfaces. In the end an analysis of a case study is carried out.

The last chapter is the conclusion of this report; and includes a technical summary along with some suggestions for future work.

Chapter 2

Terrain Modeling

Traditionally a *natural terrain* has been modeled for computer graphic visualization as collection of polygons, a specific color of each polygon representing the average altitude in 3-D. Representing the elevation at each point, graphically (as a bitmap) is a quite costly in terms of storage space.

Some other methods used include rendering the *natural terrain* as collection of B-spline or Bezier surface patches of high order. All these methods however lack either in versatility for adoption to different types of terrain or in the visual realism of the pictures created. Mandelbrot's ([Mandelbrot 82]) approach of exploiting the *self similarity* of nature, and using the random fractals for the *natural terrain* has generated great interest.

A *natural terrain* is generally characterized by randomly distributed features that are recognized by their overall properties as opposed to specific macroscopic features. The strong stochastic features of the *natural terrain* make it a good choice for application of stochastic models [Fournier 82].

2.1 Fractal Definition of the Terrain

Fractional Brownian Motion(fBm) ([Fournier 82, Saupe 88, Mandelbrot 82]) is a family of one-dimensional Gaussian Stochastic processes. In general the fBm function [Saupe 88], $V_H(t)$ is a single valued function of one variable,t such that:-

$$\Delta V \propto \Delta t^H, 0 < H < 1. \tag{2.1}$$

where H relates typical change in V, $(\Delta V = V(t_2) - V(t_1))$ to change in t $(\Delta t = t_2 - t_1)$.

If the single valued function t is replaced by coordinates x and y in a plane, to give

$V_H(x, y)$, the altitude at position (x,y) is given by:

$$\Delta V \propto \Delta r^H \tag{2.2}$$

where, $\Delta r^2 = \Delta x^2 + \Delta y^2$ is the distance between two sample points. This implies that the altitude variations between two points are proportional to their Euclidean distance raised to a factor, H. The value of H will determine the *smoothness* of the terrain. Smaller the value of H, smoother the surface.

One of the methods that uses this fBm concept, is the *Midpoint displacement* method, to be used in this work. For this the landscape is divided into a coarse grid. A grid rectangle is the fundamental modeling primitive used here. Four vertices of the rectangle are the points whose altitudes are known to us. To generate the midpoints for the edges of the rectangle, and one more vertex at the center of the rectangle (using any of the diagonals), following method is used:

$$x_m = (x_0 + x_1)/2 \tag{2.3}$$

$$z_m = (z_0 + z_1)/2 \tag{2.4}$$

$$y_m = (y_0 + y_1)/2 + y_\rho \tag{2.5}$$

The planar coordinates are generated deterministically, while the y (as we are working in XZ plane)coordinate has a non-deterministic component y_ρ along with the deterministic component. A slightly simpler version of equation 2.2, is used here to determine y_ρ

$$y_\rho = r * (\rho(x, z) \text{mod} s) \tag{2.6}$$

The r here is the distance between the two vertices (whose midpoint is being generated), ρ is a pseudorandom Gaussian number, seed tied to the x , z values of the two vertices and s is the surface smoothness controlling function (this surface roughness/smoothness parameter is also interpolated for the midpoint of the two vertices), s acts somewhat like H (equation 2.2) and controls the range of ρ

The seed of the guassian variable need to be tied down to satisfy the *consistency* criteria. There are two consistency criteria [Fournier 82] to be satisfied to get a smooth picture. The *internal consistency* is the reproductibility of the primitive at any position in an appropriate coordinate space and at any level of detail. This means that the features of the modeling primitive should not depend on its position in the space and features visible at

high resolution should be consistent with those at a coarser scale. The *external consistency* refers to the continuity properties of the adjacent modeling primitives. This is ensured by tying the seeds, so that the midpoints of, the edges shared by the adjacent primitive, will have the same values as generated from the adjacent primitive.

Each grid rectangle creates four more lower level primitives recursively, till a satisfactory level of detail s reached.

2.2 The Grid Rectangle

The largest unit in the map, on which the fractalisation is applied, is the *grid rectangle*, whose vertex coordinates depend completely on the input data. The size and shape of the *grid rectangle* not only affects the visual realism of the picture built but also the internal consistency (see 2.1) of the model.

A grid which is sparse, is much easier to store and overlay, while it has to rely on the fractal method used, to provide realism and integrity of representation. A dense grid on other hand, occupies much more memory space and is wasteful but provides more faithful reproduction of the original landscape.

In terms of the periodicity of the sampling intervals, *regular* as well as *irregular* grids can be employed. A regular grid is quite simple to construct , and easily adopts itself to the atlas-like representation in terms of *latitudes* and *longitudes*. There are however some obvious drawbacks, which are as follows:

- It is very difficult to choose the appropriate size of the grid. The size should be adjusted to the roughest terrain in the picture, at the same time, it must be highly redundant in smooth terrain.
- Important features/objects are likely to be missed, while periodic sampling on smooth flat surfaces become wasteful.
- Different applications of the DTM demand representations at different resolutions, which is not possible with a regular grid.

2.2.1 Irregular Grid

An irregular grid is designed to overcome the problems mentioned above. The sampling interval in both X and Z coordinates can vary as per the lay of the ground. This type of grid

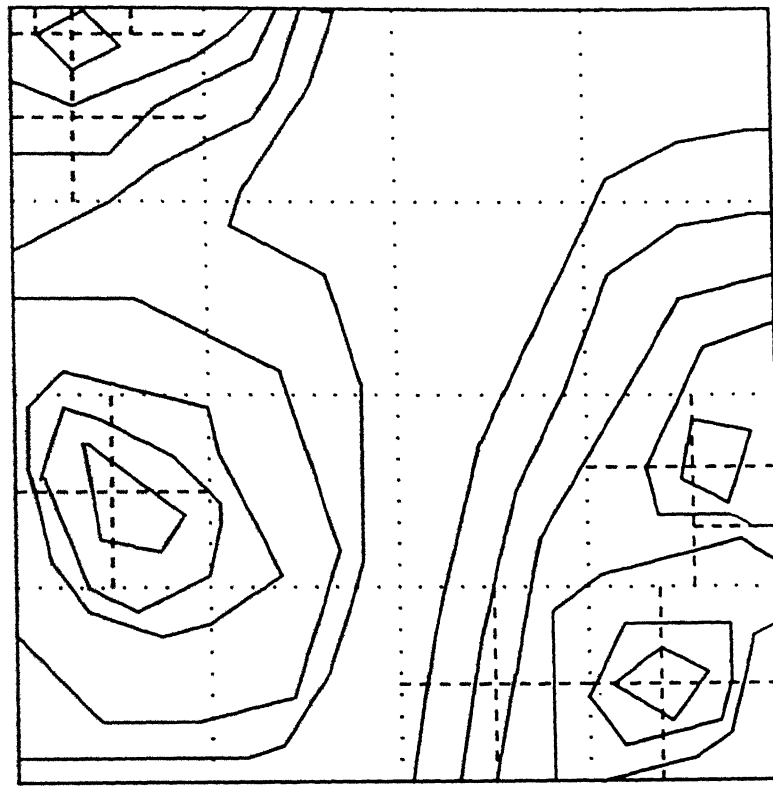


Figure 2.1: The Irregular Grid Overlayed over the regular Grid

is more difficult to create(specially as the sampling points are located randomly), though it may effect saving in terms of storage space.

The type of grid formation chosen here, is a variable density irregular grid superimposed on the regular sparse grid structure. The regular grid is used for its extendibility to conventional geographical representations. Some important features, objects and discontinuities, which cannot be represented using the regular grid, are picked up as additional *sampling points*. Each of this point now forms the central vertex in a *quad-grid*[1] which springs out of the parent *grid rectangle*. If the sampling point falls on a shared edge of the two parent rectangles, it simply splits the rectangles into two.

2.2.2 The Aspect Ratio

The method described above is likely to produce oblong rectangles in some cases, which can lead to highly unbalanced fractalisation. To avoid this condition, a limit is imposed on the aspect ratio of the rectangles, which is $(1 : \sqrt{2})$. Any rectangle exceeding this aspect ratio, is broken up in the center along the longer sides. Choice of the factor$\sqrt{2}$, ensures that a

[1] Quad-grid is a grid system with four adjacent grid rectangles.

stable state in this process of splitting the rectangles, is achieved quickly. Illustration of an irregular grid over a regular grid is in figure 2.1 .

2.3 Extraction of the Altitude Data

The terrain modeled as above can now provide reasonably accurate elevation data without having to store large amount of vertex data. The irregular grid vertex data, is the only data, which has to be stored in memory. To generate elevation at any given point the basic *grid rectangle* is first identified, a recursive fractal subdivision, is carried out, exactly in the same manner as it was done for rendering the picture. The elevation of the point is now obtained by interpolation of the elevation of the two diagonal vertices of this fractal rectangle in x and z coordinates and then averaging the two interpolated values.

Chapter 3

Path Planning Algorithm

3.1 Path Planning as State-space Search Problem

3.1.1 Graph specifications

Most of the *path planning* problems are problems of *state space search*, once a finite graph (set of nodes and arcs) is available. Nilsson [Nilsson 85] gives two elaborations of the problem as:-

- Find a path between a node s and any member of the set of nodes $\{t_i\}$.
- Find a path between any member of set $\{s_i\}$ and any member of set $\{t_i\}$.

The *finite graphs* are specified in two ways, *explicitly* and *implicitly.*

For an explicit specification, the nodes and arcs (with associated costs) are explicitly given, (say) in a table.

In an implicit specification, a finite set $\{s_i\}$ of nodes is given as start nodes. In addition a successor operator Γ is given, that can be applied to any node, to give all the successors of that node and the cost of associated arcs. In state-space terminology, the successor operator is defined in terms of the set of operators applicable to a given state description. The application of Γ to the members of $\{s_i\}$,to their successors and so on, then generates the graph implicitly, defined by Γ and $\{s_i\}$.

Most of the methods of *path planning* discussed in 1.3 first create a finite graph and subsequently use this *explicit* graph for solving the state-space-search problem. This method is quite compact and each problem partition remains independent of each other..

In this work, an *implicit* specification is used instead. Apart from the saving accrued, in the storage space required for the *explicit* graph, main motivation to choose this specification, was the availability of the information generated during the heuristic search, which

can be used for graph generation decisions. This information can thus be exploited to limit the wasteful, wavefront like generation of the graph. The heuristics required for this exploitation can be easily incorporated in the set of operators Γ . These heuristics and the graph generation method is described later in this chapter.

3.1.2 Definition of problem as state-space search

Rich [Rich 83] gives steps to define a problem precisely as a state-space search problem. Following these steps, we define the problem as follows:-

- The problem space is the DTM representing the area, while the search space is defined by the *graph* to be generated.
- The *initial* state is the location in x,z coordinates of the START point.
- The GOAL state is the location in x,z coordinates of the GOAL point.
- The path follows following rules:-
  1. An optimal path using a path cost evaluation function is to be followed.
     The arc cost consists of weighted sum of costs due to the Euclidean distance, the gradient between the two nodes and cost of traversal through high cost regions(thematic polygons) if any.
  2. Cost of traversal through *forbidden regions* is infinity.

3.1.3 Search Methods

Search methods fall in two categories ([Rich 83]).

- Blind Search.
- Heuristic Search.

The commonly used blind search techniques include *depth first search* and *breadth first search*. The heuristic methods use certain information about the nodes that aids the search, to reduce and consequently speed up the search. *Best first search* is an example of a heuristic search method. These methods are explained in detail in [Rich 83] and [Nilsson 85].

3.2 The A* Algorithm

The algorithm used here is a modified A* algorithm which is an implementation of *best first search*. The A* algorithm is spelt out as below [Rich 83].

Each node in the problem space will contain, in addition to a description of the problem state it represents, an indication of how promising it is, a parent link that points back to the best node from which it came, and a list of nodes that were generated from it. The parent link will make it possible to recover the path to goal, once the goal is found. The list of successors will make it possible, if a better path is found to an already existing node, to propagate the improvement down to its successors.

We will need to use two lists of nodes:

- *OPEN*—nodes that have been generated and have had the heuristic function applied to them, but which have not yet been examined(i. e. , had their successors generated). This list is really a priority queue in which the elements with highest priority are those with the most promising value of the heuristic function. Standard techniques for manipulating priority queues can be used to manipulate the list.
- *CLOSED*—nodes that have already been examined.

We will also need a heuristic function that estimates the merits of each node we generate. This will enable the algorithm to search more promising paths first. Call this function f' (to indicate that it is an approximation to a function f that gives true evaluation of the node). For many applications it is convenient to define this function as the sum of two components, which we shall call g' and h'. The function g is the measure of the cost of getting from the initial state to the current node. Note that g is not an estimate of anything: it is known exactly to be the sum of the costs of applying each of the rules that were applied along the best path to the node. The function h' is an estimate of the additional cost of getting from current node to the GOAL state. This is the place where knowledge about the problem domain is exploited. The combined function f' then, represents cost of getting from initial state to the GOAL state along the path that generated the current node. If more than one path generated the node then the algorithm must record the best path. It is important that g be non-negative else paths that traverse cycles in the graph will appear to get better as they get longer.

The algorithm statement is as given below.

The A Algorithm*

1. Start with OPEN containing only the initial node. Set that node's g value to 0, its h' value to whatever it is, and its f' value to h'. Set CLOSED to the empty list.

2. Until a GOAL node is found, repeat the following procedure: If there are no nodes on OPEN report failure. Otherwise, pick the node on OPEN with the lowest f' value. Call it BESTNODE . Remove it from OPEN . Place it on closed. See if BESTNODE is GOAL node. If so exit and report a solution (generate the path up to the BESTNODE if we are interested.) Otherwise generate the successors of BESTNODE but don't set BESTNODE to point to them yet. (First we need to see if any of them have already been generated.) For each SUCCESSOR, do the following:

   (a) Set SUCCESSOR to point back to BESTNODE. These back links will make it possible to recover the path once the solution is found.

   (b) Compute g(SUCCESSOR), = g(BESTNODE) + cost of getting from BESTNODE to SUCCESSOR.

   (c) See if SUCCESSOR is the same as any node on OPEN (i.e it has already been generated but not processed). If so, call that node OLD. Since this node already exists in the graph, we can throw SUCCESSOR away, and add OLD to the list of BESTNODE's successors. Now we must decide whether OLD's parent link should be reset to point to BESTNODE. It should be if the path we have just found is cheaper than the current best path to OLD (since OLD and SUCCESSOR are really the same node). This is done by comparing their g values. If OLD is cheaper then we do nothing. If SUCCESSOR is cheaper, then reset OLD's parent link to point to BESTNODE, record the new cheaper path in g(OLD), and update f'(OLD).

   (d) If SUCCESSOR was not on OPEN, then see if it is on CLOSED. If so call the node on CLOSED OLD, and add OLD to the list of BESTNODE's successors. Check to see if the new path is better just as in step 2. (b), and set the parent link and g and f' values appropriately. If we have found a better path to OLD, we must propagate the improvement to OLD's successors. This is a bit tricky. OLD

points to its successors. Each SUCCESSOR in turn points to it's SUCCESSOR and so forth, until each branch terminates with a node that is either still on OPEN or has no successors. So to propagate the new cost downwards, do a depth first traversal of the tree starting at OLD, changing each node's g and f' values, terminating each branch when you reach either a node with no successors or a node to which equivalent or better path has already been found. This condition is easy to check for. Each node's parent link point to it's best known parent. As we propagate down to a node, see if it's parent points to the node we are coming from. If so continue the propagation. If not, then compare the g values of the two parents through the two different paths, stop propagation if the current parent is still better. Else reset the parent and continue the propagation.

(e) If SUCCESSOR was already on either on OPEN or CLOSED, then put it in OPEN and add to the list of BESTNODE's successors. Compute f'(SUCCESSOR) = g(SUCCESSOR) + h'(SUCCESSOR).

3.3 Graph Generation and Implementation of A*

3.3.1 Uniform versus Non-uniform Regions

In the earlier works of Mitchel, Rowe and Richbourg (see 1.3), certain facts/assumptions regarding local behavior of optimal paths were exploited to create a finite graph. Some of them are listed below.

1. Shortest Paths are piecewise-linear paths, bending only at points where they cross an edge, or pass through a vertex. This follows the fact that each individual weighted region is assumed to be *uniform.*

2. The whole map space is considered to be homogeneous, interspersed with discrete but homogeneous regions/obstacles.

If the cost of climbing a *gradient* on these path is included, (2) above cannot be assumed. The weighted regions do not remain *uniform* in cost. The thematic costs are to be applied uniformly over and above the *gradient* cost[1] Consequently, assumption 1 also does not hold as now we are dealing with non-uniform regions.

[1]This implies that passing through a forest on the slopes of a mountain is more costly than passing through a forest of the same density in plains—ask any trekker!

The problem of non-uniform region makes the *path planning* more interesting and complex, as we have to deal with non-linear curves in three dimension [Mitchel 88]. The infinite problem space of the map has to be reduced to a finite graph...heuristically. This work makes an attempt to propose some of such heuristics.

3.3.2 Rules for Graph Generation

The graph contains two basic units; *nodes* and *arcs*. Some assumptions regarding these are as follows:

- The smallest unit of a path is an *arc* . The length of an arc should be as small as possible for optimality, however too small an arc-size may increase size of the graph. Call this optimal arc length as STEP.
- A *node* is a vertex on either end of an *arc*. The START and GOAL nodes are special kinds of nodes.
- *Gradient* over an arc is assumed to be uniform and is the slope of the arc $((y_2 - y_1)/(x_2 - x_1))$.

The lowest level procedure in the graph generation is the generation of successors from any node using a Γ operator (refer 3.1.1). The procedure is as follows;

1. The *central axis* is defined to be the euclidean direction(bearing) towards the GOAL. The first SUCCESSOR is the geographical point STEP distance away from the node, on this axis.
2. Number of successors are now generated in arc of 180 degrees (90 degrees on either side of the axis)[2].
3. The maximum number of successors will be defined by the inter–arc angle, which is constant for all arcs. The value of this angle will affect the density of the graph.

These three rules of thumb create a graph in a manner somewhat similar to the wavefront approach on a pixel map described in [Mitchel 88]. To avoid inheriting the disadvantages of the wavefront method [Rowe 90], some constraints are applied to generate a

[2]The limit of 180 degrees is imposed to retain the directionality of the graph and avoid loops.

more directional, less wasteful and unbiased graph. These constraints exploit the additional information regarding arc-costs, generated during the search (and hence *implicit* graph specification). The constraints developed through experimental learning, are mainly directed towards solving problems containing high cost due to the *gradient*. They are easily extendible to not–so–high cost weighted regions also (additional heuristics are required for high cost and forbidden regions, which are explained later in this chapter). The constraints are given below:

1. Minimum three (central and two on the sides) successor arcs are to be generated, except when one or two of them fall outside the map space.

2. From the three links a cross-sectional study of the costs returned is carried out.

   The actions to be taken are:

   (a) If one of the the arc-costs is below a threshold value, no more spreading out is required as we are right on track. Abort further generation of successors.

   (b) If the central arc is the of a higher cost than the lateral ones, either sides promise an arc below the threshold. Generate two more arcs (one on either side) with the required angular separation from the lateral ones.

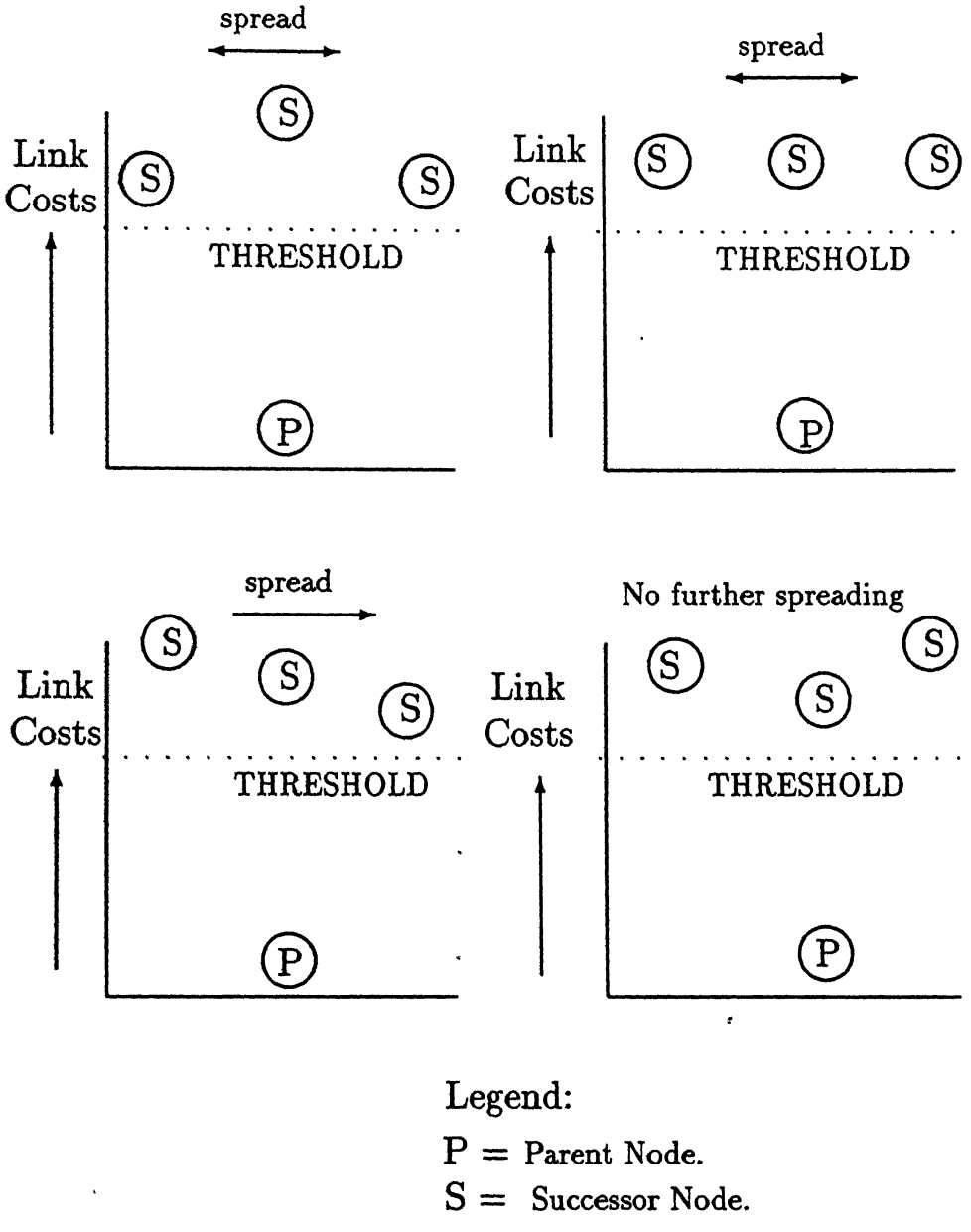
   (c) A flat cross section (all link costs more or less same) do not indicate any promising side, but cannot be rejected. Treat it the same way as (b) above.

   (d) An inclined cross-section (Highest and lowest cost arcs at the extremes) indicate a single promising side. Generate only one more arc on the promising (lower cost) side.

   (e) A 'V' shaped cross section (extremes with high cost with respect to the central one) indicate lack of promise on either side. Abort further successor generation.

3. After the initial three, none, one more or two more arcs are generated. Continue processing the graph in the same way, applying the same test conditions, only the extreme arcs recently generated and the original central one are of interest. The spreading out to continue till either it reaches 180 degrees or satisfies an ABORT condition.

The generation of successors for conditions 2(b) to 2(e) is illustrated in figure 3.1 .



Legend:

P = Parent Node.

S = Successor Node.

Figure 3.1: Heuristics for Successor Generation

An assumption implied here is that, the cross-section between two adjacent arcs is continuous. The discontinuity, in practice, will be over an insignificantly small geographical area.

The node just short of the GOAL is treated in a different way than the default method described above. (Otherwise the goal may be overshot and graph generation will remain in oscillations). This small modification is:

- When any node generated is within STEP distance of the GOAL, only one successor (The GOAL) is attributed to this node.

3.3.3 Implementation of A*

The method described in this chapter implements the A* algorithm (with some modifications).

The issues involved in the implementation are:

- The heuristic function h'.
- The Evaluation function generating g.
- The GOAL state.

The GOAL state out of these is quite obviously the geographical location of the GOAL. The heuristic function h' is generated by calculating the Euclidean distance from any *node* to the GOAL. The cost of traversal of the agent over this distance on an ideal surface gives the value for h'. The evaluation function calculates the arc cost. This function is the weighted sum of the following quantities.

- Cost of traversal due to the distance(arclength) on an ideal surface.
- Penalty due to a gradient. This penalty is proportional to the square of the *gradient*. This means that the cost against the gradient rises parabolically—a rule learnt through experience of driving/walking in hills.
- Penalty/unit distance of traversal through a weighted region. Applied separately for each polygon of each theme.

The weightages of each cost, in the cost-evaluation equation, depends on the users perception of the problem and the type of the mobile agent.

3.4 The Staged Search

The areas containing high cost thematic regions have to be treated slightly differently. This follows a intuitive approach of skirting around, or diverting away from these regions. Heuristics of a *staged search* explained in [Nilsson 85] are employed for this purpose. These are described below:

- A list of all high cost/forbidden thematic polygons is maintained. The polygons have to be convex and disjoint. Some preprocessing before the actual run has to ensure this. The non-convex, non-disjoint polygons are split into number of convex polygons. Method for splitting a non-convex polygon into convex polygon is explained in[Rogers 85]. A convex cover of two(or more) jointed regions can represent the polygons for the purpose of this technique. Doing so will also reduce the number of polygons and hence the complexity.

- The actual problem of finding an optimal path between the START and the GOAL is to be divided into number of subproblems, wherein the search procedure will be successively applied sectionally(between a TEMPSTART and a TEMPGOAL). The method of division and subsequent assembly of the problem is as follows:

  1. Initially assign GOAL to the TEMPGOAL and START to the TEMPSTART.
  2. See if any polygon in the high-cost polygon list (let us call it AVOIDLIST) is intersecting the Euclidean straight line path between the TEMPSTART and GOAL. If no intersection is reported, run the *path search* between the two and store the sectional, optimal path in a separate linked list.
  3. If one or more intersections are reported, take the nearest edge of the nearest polygon (in the direction of the GOAL, from the TEMPSTART) for further processing. We now have two vertices of the edge, forming the TEMPGOAL for two subproblems starting from the TEMPSTART. We proceed in a *depth first* manner for each subproblem as follows:
     (a) Run the *optimal path search* between the TEMPSTART and the TEMPGOAL. store the optimal sectional path as in 2 above.
     (b) ASSIGN the previous TEMPGOAL as TEMPSTART. Check if the polygon containing the edge is self-intersecting the direct segment between the

TEMPSTART and the GOAL. If no intersection is reported skip the next step and go back to step 2.

(c) As the polygon is intersecting, it has to be skirted along the perimeter. This is done by selecting the next vertex in a clockwise or anticlockwise (the aim is to move away from the other vertex of the original edge selected in 3 above) manner as the TEMPGOAL. Go to step 3(b) for progressing along the perimeter till the polygon is not self-intersecting.

4. At this stage we have one[3] or more of sectionally optimal paths. To generate the overall solution, we join the lowest cost, contiguous sections between START and the GOAL. The g value cost for a *node* is retained between the sections. This allows us to report the path cost to the GOAL without any further calculations.

- Some tricks to avoid wasteful generation of sectional paths can be employed. It would be a waste to regenerate a path, even if the ancestors of the current TEMPSTART are different(but the TEMPGOAL is the same). If the physical location of TEMPSTARTs is the same(vertex), and subsequent path has already been generated, further progress towards the GOAL can be abandoned (due to the depth first nature...the GOAL has already been reached along the path).

The staged search is illustrated in figure 3.2 .

3.5 Complexity and Efficiency

Mitchel ([Mitchel 88]) reports the worst case time complexity of a shortest path algorithm like the A* to be of the order of $O(n \log n)$ where n are number of vertices. For that matter in an obstacle avoidance problem it is reported to be of the order of $O(E + V \log V)$ where E are number of edges and V are number of vertices. The A* implemented here should be of the same order, $O(n \log n)$ where n are the number of nodes.

The real bottle-neck is in the *graph generation*. The most basic *visibility graph* (VGRAPH) approach for obstacle avoidance [Lozano 79] has to compare every pair of vertices for checking the condition of intervisibility. There are $O(n^2)$ such pairs, n being the number of obstacle vertices. The algorithm for generating the graph takes $O(n^2 \log n)$ to compute

[3] If the number of sections is only one then it is the final solution.

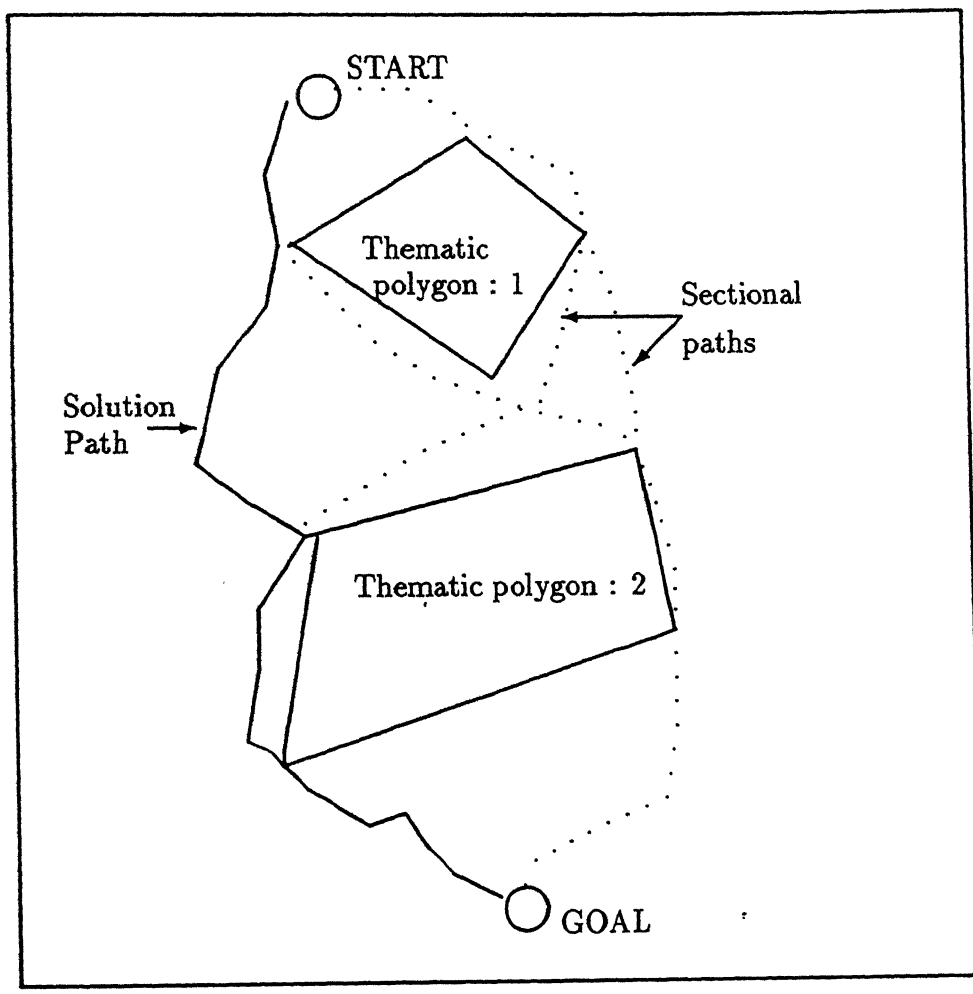


Figure 3.2: Staged Search: Showing the Sectional and Final solutions

the VGRAPH ([Mitchel 88][Lozano 79])[4]. The algorithms for roads, obstacles and rivers of both Mitchel and Rowe(([Mitchel 88], [Rowe 90]) report a worst case time complexity of $O(n^2 \log n)$.

The proposed method propagates somewhat similar to (actually a special case of) wavefront type approach(on a weighted region, digitized grid map) . The wavefront method as described by Mitchel([Mitchel 88]), runs in overall worst case time complexity of $O(n \log n)$. This cost is suggestive of the time complexity of our method, though analysis in terms of time complexity has not been carried out, due to the shortage of time and problems in determining the time complexity in an implicit method of graph generation.

The costs associated with the successor generation heuristics will also have to be taken into consideration while calculating the overall time complexity.

The overheads due to *staged search* are more than compensated by the wasteful search it saves. Actually additional cost due to the separation and aggregation of sections is limited to the order of $O(P)$, where P are number of polygon in the AVOIDLIST.

There is a tradeoff between *efficiency* and *optimality* in a heuristic search ([Rich 83]). A search with the heuristic function underestimating the actual path cost, is bound to give optimal results, while an overestimated heuristic function can reach the GOAL much faster, but it's optimality cannot be guaranteed ([Nilsson 85],[Rich 83]). The emphasis here has been on *optimality*. The hueristic function h' takes the lowest possible path cost, thus uniformly underestimating. However a formal study of these aspects too needs to be carried out.

[4]improvements on this algorithm has resulted in algorithms of $O(n^2)$([Mitchel 88]).

Chapter 4

Software Development

The software for solving and testing the *path planning* method over a *natural terrain* was developed using C language and HP-STARBASE graphics routines over a HP 98731 workstation graphic terminals. Two independent modules of the software were developed, to be used together or independently. First module dealt with purely the computer graphic rendering of the given map of the terrain and the second module dealt with running the *path planning* algorithm for a given problem in this map-space. A user interface interlinked the two with simple *system* calls. Some of the important issues involved in development of the software are discussed in this chapter.

4.1 Graphic Issues

The STARBASE graphic library used here provides a whole range of functions and primitives for realistic rendering of three dimensional models. Various important graphics issues that were tackled are discussed subsequently.

4.1.1 Rendering the Fractal Polygons

The Fractal Triangles

In 2.1 method for generation of a fractal rectangle of appropriate resolution, was discussed [1]. These fractal rectangles so created, however cannot be directly sent as a rendering primitive as the rectangles may be non-planar. The rectangles are split into two triangles along a diagonal. This method also has an advantage of allowing us to color the two parts of the original rectangle, with different colors, improving the color-mixing for visual realism. The triangles are rendered using the starbase primitive of *polygon3d.*

[1] The lowest possible resolution is the *pixel* level.

Coloring and Shading

Starbase provides various options for coloring/shading a polygon. The colormap can be set to MONOTONIC, NORMAL or FULL. A FULL colormap (used here) provides 24 planes of buffer for a colormap. The hue and saturation of the color can be adjusted by setting the RGB (Red, Green and Blue) values appropriately, within a range from 0.0 to 1.0.

Three types of *shading* methods are provided by STARBASE; Polygonal shading, Garaud shading and Phong shading[2]. Here the simple polygonal shading is used to avoid adding complexity, due to calculations in formulating the light equations for the vertices, required by the other two methods.

The coloring decision method is similar to one illustrated in [Starbase] The method is described as follows:

- The range of the altitude, encountered on the map is broken up into three or four subranges. The division is based on the aerial appearance of the landscape due to change in the *surface cover*, as the altitude changes. A single RGB value is associated with each such subrange. E. g. the lower planes have the color of grasslands, the foothills a greenish brown, the mountain ranges stark blackish brown while the snow covered tops are colored shining white.

- Only the regions away from the boundaries of another adjacent subrange are colored by the allotted subrange RGB value. The boundary regions are colored in a mix of the two involved subranges.

- The mixing effect is created by randomly coloring the polygons generated in the two different colors. The randomness is controlled so has to have a proper ratio of mix of the two colors, duly interpolated over the distance from the boundary. We thus get an effect of gradual and continuous change of the surface cover over the range of the altitude.

4.1.2 Three Dimensional Rendering and Perspective Effects

An imaginary *camera* is provided by STARBASE to apply the transformations, required for realistic effect in 3-D. Various parameters of the *camera*, such as; camera reference point, viewing angle, camera location, etc can be defined to obtain the required view. Position of

[2] see [Rogers 85] for details of these methods.

the camera and the light source effect the shading of the polygons. *Depth cueing* creates effects of 'fogginess' or 'fading light' for further realism. STARBASE uses a *strip Z-buffer* (a method which divides the picture into strips and then applies Z-buffer) algorithm for hidden surface removal. Though the computational and memory overheads of any hidden surface removal are considerable, the strip Z-buffer attempts to reduce this by dynamic allotment of memory to various strips.

4.1.3 Representation of Solution Paths/Areas

The solution arcs of the *path planning* problem are overlaid on the terrain picture using simple line drawing primitives. A piecewise linear path is seen as a result. A smoother look can be easily created by using B-spline or any other curves. However this does not provide a faithful reproduction of the solution, as generated by our method.

4.1.4 Thematic Polygons and Clipping

The thematic polygons (weighted regions) are rendered as hollow polygons with a unique color representing a theme. When an arc on the solution graph intersects a polygon, the length of the intercept determines the cost accrued due to the polygon. A two dimensional line clipping algorithm, the Cyrus-Beck algorithm, is used to determine this (refer [Rogers 85]).

In the staged search method proposed in 3.4, the nearest intersecting edge from the START is determined using the same clipping algorithm.

4.2 Database and Datastructures

4.2.1 The input Database

Database issues are one of the prime issues in development for the GIS. However these issues were not dealt in detail for the purpose of this work. The input database for the programs is organized in three different data files as follows:

- *The map and Grid Datafile.* This file acts as a digital storage for the mapsheet. First entry of the mapfile contains the minimum and maximum limits(the range) in the planar coordinates for the whole map. Rest of the mapfile contains data about each individual grid rectangle; a line for each such rectangle. The x,y,z and the surface

roughness value s,(refer 2.1) for each vertex of the rectangle are stored in the same sequence. This file is used mainly by the *path planning* subroutines.

- *Thematic Polygon Information file.* The hierarchical datastructures of the thematic polygons are stored in this file. The file contains entries as follows:

  1. Number of themes.
  2. For each theme, the number of polygons in the theme and the weightage the theme receives in the cost evaluation function.
  3. For each polygon in the theme, it's cost and number of vertices.
  4. For each vertex in the polygon, the x,z coordinates (altitude is unimportant for thematic polygons.

- *Camera and color setting file.* This file contains the various camera settings(refer 4.1.2), the color codes (refer 4.1.1). Other settings dealing with realistic rendering such as clipping planes, planes and colors for depth cueing are also set through this file.

4.2.2 Data structures in the program

Most of the datastructures used in the programs can be physically related to geometrical objects such as a point, arc, rectangle or a polygon. Other structures are mainly pointers to such structures, to create and manipulate various linked lists, used for searching and backtracking the solution. Few of the important and globally used datastructures are discussed here. The routines used for graphic rendering and search computations, both use datastructures which are aggregated over a base structure 'Point'.

The fields of the structure *Point* determine the physical location in X,Z coordinates of the point on the map, it's altitude (Y coordinate) and the surface smoothness parameter,s (see 2.1). A simple array of four *Points* define a fractal/grid rectangle.

A *node* in the graph to be generated has a datastructure to suit the *path planning* algorithm. One of the field obviously being a generic *Point*, rest of the fields are as follows:

- *g value, f' value and h 'value.* These values are required by the A* algorithm (see 3.2) and are represented as single floating point quantities.

- *Parent, next, prv.* These are pointers to other *nodes*; Parent pointer is used for backtracking a solution, and for recursive updating improvements in the g values. Other two pointers are used for list manipulation of a doubly linked list.

- *Nodeid.* The *node* is identified by this unique integer.

- *Noofsucc.* This field gives the numbers of successors a node has. Valid only when a *node* goes on to CLOSED.

- *Dir, nodeslope.* The *bearing* to the GOALor TEMPGOAL is denoted by *dir* field. The *nodeslope* is used only for diagnostic purposes, denotes the *gradient* of the *node* from it's parent.

The data structure required to represent thematic polygons, keeps in mind their role in a staged search(see 3.4). The vertices of the polygons and the inward normals to the edges (required for the Cyrus-Beck clipping algorithm), are represented as arrays of the structure *Point*(Only the X Z coordinates being important), in the structure *Pollist* (indicating list of such polygons). The no of vertices of the polygon, and indices in the array of the vertices forming the clipping edge are stored as integer fields in this structure. A pointer *pnext* assists manipulation of a linked list of polygons. Another floating point parameter, *mindist* is used for resolution of the nearest clipping edge of a polygon nearest to the TEMPSTART.

4.3 User Interface

The user interface of the *path planning* problem has to be application dependent. The interface has to cater for two broad aspects of the problem:

- *Problem Input.* This involves defining the problem environment and the problem data set. The problem environment consists of the input *map* and the *sampling points* for the DTM generation. It also consists of the themes and the thematic polygon data. The *problem dataset* consists of the coordinates of START and GOAL points and some variable parameters such as the STEP size and the weightages of the *gradient* and *distance* costs.

- *Results and Analytical information display.* Apart from displaying the actual optimal solution path, large amount of information, for diagnostic and analytical purposes, can be extracted from the programs. The information includes the following:

1. The path costs of optimal and second best path (in case of a *staged search*) along with the trace of the second best path.
2. Compass bearings of each node on the optimal path.
3. A cross-sectional link cost and gradient profile of the solution path.
4. Values of parameters used as measures of performance. The parameters include total path length, Number of nodes generated by the technique, time taken, *penetrance* and *effective branching factor* (see [Nilsson 85] for definition of these two), etc. .

4.3.1 Problem Input

Programs for generation of the DTM and *path planning* require three input files as discussed in 4.2.1.

The first file *map and grid data* file is created using a small program. A user has to interactively input the X,Y,Z coordinates and the surface roughness values for all the coarse grid vertex points and the additional *sampling* points. This data is stored in a temporary file. The program generating the DTM takes over, creates the irregular grid, and stores data pertaining to this irregular grid in the *map and grid data file*, which is used both for generating the DTM and later in the *path planning* program. The coordinates for the vertices of the coarse grid and the sampling points have to be picked up manually from the contour map. In any case this is a one time exercise as the file corresponding to a single mapsheet or a map rectangle can be preserved, archived and invoked by the map-index whenever required.

Another small program facilitates creation of the *thematic polygon information file*. Data as given in 4.2.1 is obtained interactively from the user, by this program, inward normals calculated (method explained in [Rogers 85]) and the file created. Again archiving of the polygon files is possible, specially in a GIS scenario.

The *camera and color* setting file, at present is created manually, by the user.

The whole process of problem input can integrated in a compact, menu driven system. This is a suggestion for future work (a mini-project or a term project).

4.3.2 Results and Analytical Information Display

Presently the user gets a display of the optimal solution (and also all the branches generated), overlaid on the DTM, in form of simple line segments in different colors. In addition, the final path cost, the *nodeid*'s of the nodes forming the solution and the total path length, are available directly on the terminal for a quick assessment of the solution.

Results pertaining to the performance measurement parameters, are stored separately in a data file for further analysis.

A graph plot giving cross-sectional view of the arc-costs and gradients on the optimal path, and compass bearings of each node provide a readymade route-chart for the agents setting out to perform the move. This plot is created in a different window/view-port, simultaneously with the DTM, and should form part of a menu driven, window based system suggested in 4.3.1.

4.4 Simple Cases—Trial Surfaces

A *natural terrain* surface is known for it's strong stochastic properties. However breaking down the *natural terrain* surface feature by feature, can yield a collection of well defined geometric surfaces.

During the development of the *path planning* method described in the previous chapter, some of such surfaces were used as trial surfaces for carrying out trials of new ideas and testing validity of methods developed. As the methods developed were purely heuristics in nature, it was difficult to access their success quantitatively, or measure their performance on any standard scale. Visual inspection of results was the only method available. Lack of human experts in this field at hand, for critical examination using the visual method, prompted use of simpler, easily identifiable surfaces to run the algorithms and heuristics.

Three surfaces, each defined easily through simple equations were used as trial surfaces. The surfaces and important indications obtained through them are given subsequently.

4.4.1 A Conical section

A conical section was generated using polygonal primitives, by rotation by 360 degrees of an inclined line segment in XZ plane. The cone was placed on a plane 'platform'. The equation of the cone enables assigning altitude for any point on the cone. Various trial points were given as START and the GOAL to try out various versions of the *path planning*

heuristics. Some observations and indications obtained from trials on this surface are as follows:

- On encountering an inclined *cross-section* (case 2(d) in 3.3.2), it is wasteful to try and spread out further towards the high-cost end of the cross-section.
- If all the successors return more or less equal high costs (case 2(b) in 3.3.2), it's wasteful to spread on both sides. The optimal path is likely to be branched off from an earlier node.

4.4.2 A $\sin R/R$ Surface

This surface has high amplitude peak in the center and peaks and crest with amplitudes reducing towards the periphery. The surface is created by first drawing a curve following the equation $y = \sin 2\pi x/2\pi x$ and then rotating it in 360 degrees. On applying various trial problems on this surface following observations were recorded:

- A plateau like cross-section (see observation 2 above) may still hold a promise by spreading on either side, as a lower cost area may lie a few degrees away, like a crest after a wave. This was contradictory to the earlier observation from the conical section.
- In a valley like cross section (see 2(e) in 3.3.2), further spreading is unnecessary as the path is likely to proceed mostly along the central successor.

4.4.3 A $\sin\theta$ Surface

The $\sin\theta$ surface looks like a corrugated sheet of metal. This surface was generated using the equation $y = \sin 2\pi x$ and then extending the curve along the Z direction. This surface consolidated most of the observations above, however failed to give any important indications. As such it is difficult to visualize optimal paths if the START and END points lie across the corrugation.

All the curves above give a strong indications that the optimal paths cover most of the distance along near-constant-cost curves(instead of zigzagging along different cost gradients). This may be due to the strong macroscopic features of the test surfaces. A *natural terrain* gives different results. This surface dependent nature of the heuristic pruning

1. Total number of nodes opened during the search.

2. Length of the optimal path as against the straight line distance between the START and the GOAL.

3. The final f value of the optimal path.

4. Time taken to reach the solution.

INPUT PROBLEM							
Prob. No.	START		GOAL		STEP size	Staged Search Enabled?	No. of Polygons Encountered
	X	Z	X	Z			
1	8.0	4.5	2.0	6.0	0.4	Yes	0
2	8.0	4.5	2.0	6.0	0.4	No	0
3	3.2	4.5	3.5	2.5	0.2	Yes	1
4	3.2	4.5	3.5	2.5	0.2	No	1
5	7.5	5.3	2.0	3.4	0.3	Yes	2
6	7.5	5.3	2.0	3.4	0.3	No	2
7	8.0	4.5	2.0	6.0	0.4	Yes	2
8	8.0	4.5	2.0	6.0	0.4	No	2
9	6.0	7.5	6.0	3.0	0.4	Yes	1
10	6.0	7.5	6.0	3.0	0.4	No	1

RESULTS OBTAINED						
Prob. No.	Aerial Length	Optimal Path Length	Path cost	Total Nodes in graph	Maximum slope	Time (in secs)
1	6.18	6.39	53.22	1103	-0.85	0.06
2	6.18	6.39	53.22	1103	-0.85	0.06
3	2.02	3.47	38.58	2353	-0.78	0.07
4	2.02	2.60	38.15	2315	-1.12	0.06
5	5.82	7.01	73.83	10542	-1.436	0.41
6	5.82	7.30	71.15	12980	-1.016	0.42
7	6.18	7.34	61.13	2713	0.628	0.10
8	6.18	7.49	62.83	7804	0.673	0.42
9	4.53	5.21	42.81	197	-0.399	0.03
10	4.53	4.59	54.46	1872	-0.22	0.09

Table 4.1: Inputs and Results for Some Sample Problems

Photographs in figures 4.1 to 4.4 display some of the examples studied. Photograph 4.1 and 4.2 contain the full screen display of a problem along with a plot in a different view port as described in 4.3, the first one uses the *staged search* heuristics while the second

method is one of the problems for designing standard heuristics applicable on all types of surfaces.

4.5 Case Study : An Example

For realistic trials of the *path planning* method for move over *natural terrain*, a geographical area with a good mix of hilly terrain, plain grasslands, thick forests and open fields was selected. This area pertained to Survey of India mapsheet No. 46 N/11. The rectangular area lies between latitudes 75 degrees 35 mins to 75 degrees 40 mins and longitudes 22 degrees 20 mins and 22 degrees 25 mins. The rectangle selected was 8.6 kms in length(Along X axis) and 9.2 kms(along Z axis) broad. The values in kms were used directly for the DTM, as world coordinates. The altitudes in this region varied between 300 to over 800 meters. The altitude (Y coordinate) was scaled down by a factor of 100 for use as world coordinates.

A coarse grid of 16 grid rectangles(4 by 4) was overlaid on this area. Each grid rectangle thus had a length of 2.15 kms and a breadth of 2.30 Kms. Number of additional points were created to increase the sampling frequency in rougher areas and also to represent non-vertex points which dictated the local lay of the ground(see 2.2.1). These points were used in creation of an *irregular grid* consisting of 179 grid rectangles. To assign the surface smoothness parameter a painstaking visual method was employed. In this, total number of contour lines found on a diagonal 1km segment (keeping the vertex in question at the center), were counted. This count was scaled up by some factor to eliminate fractions less than 1. The surface roughness values ranged from 1 to 1200, for each vertex of an irregular grid rectangle.

The coordinates and surface roughness parameter values for the irregular grid were stored in a data file (item 1 4.2.1) and a DTM as given in chapter 2 was created.

Large number of trials of the *path planning* method were carried out, for different sets of thematic rectangles, different sets of START and GOAL points and different weightages for the themes, gradient cost and distance cost. Some trials were conducted with exclusion of the *staged search* (by running the basic heuristics of graph generation in a *brute force manner*, allowing intrusion into the forbidden areas with very high penalties). The results so obtained were compared with those obtained by inclusion of the *staged search* heuristics. The analysis of results for some of these problems is tabulated in 4.1 . Parameters chosen for comparison are as follows:

photograph pertains to the same problem run without using these heuristics (see problems 7 and 8 in teble 4.1). Photographs 4.3 and 4.4 display a close up view of another such problem (problem Nos. 5 and 6 in table 4.1).

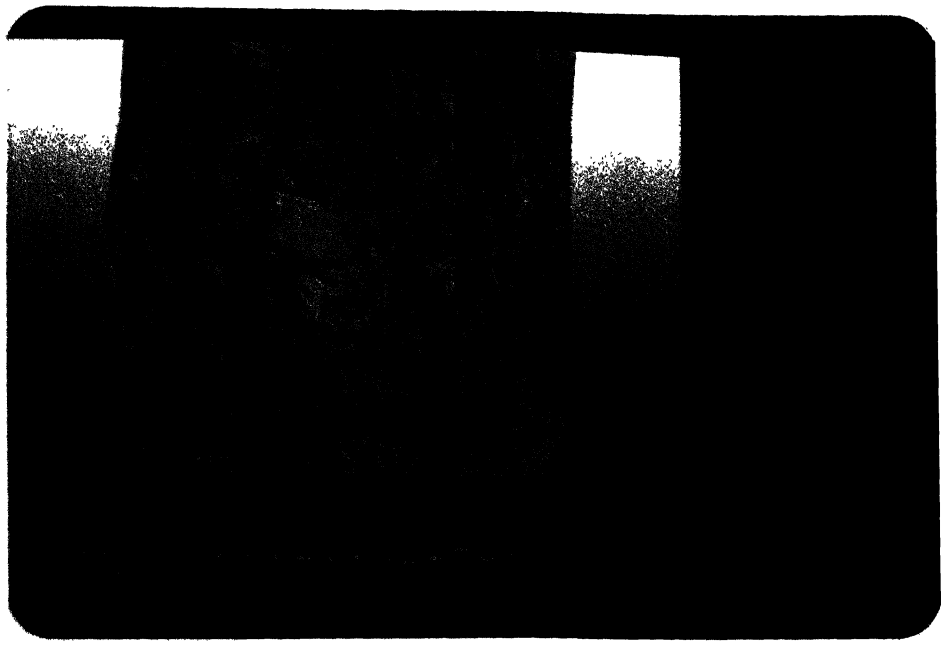


Figure 4.1: Example of Path Planning using the Staged Search Method along with the Path costs and Bearing Plot

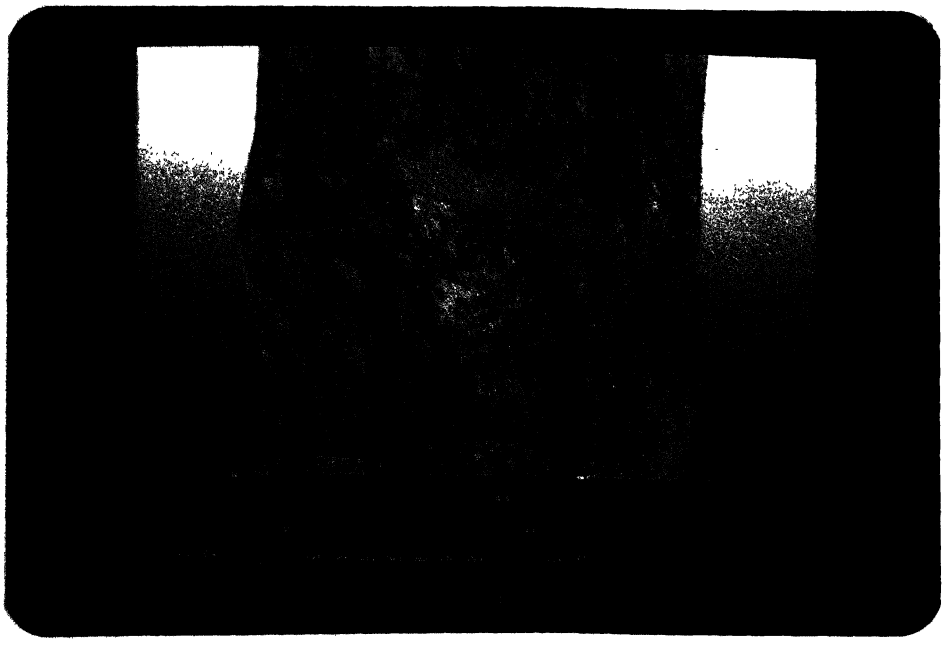


Figure 4.2: Example of Path Planning without the Staged Search for the Earlier Problem

Figure 4.3: A close up of a Path Planning Problem using Staged Search

Figure 4.4: A close up of a Path Planning Problem without the Staged Search

Chapter 5

Conclusion

5.1 Technical Summary

A new method for *path planning* on *natural terrain* was proposed in this paper. The methods for *path planning* hitherto used, projected a natural (or outdoor) terrain surface as a collection of discrete homogeneous regions with the traversal costs per unit distance changing only at the boundaries of the regions or on crossing a linear obstacle.

The method proposed here accounts for a non-homogeneous, non-discrete rough surface, implying imposing higher penalty costs for traveling over a steeper gradient, in addition to the inherent weighted costs of the thematic regions.

A *staged search* method is also proposed, which uses the vertices of the high cost polygons as the temporary bounds, and suggests a way of skirting around such polygons, likely to intercept potential optimal paths. The *staged search* was found particularly efficient as compared to a START-to-GOAL *brute force search*, when large sized polygons with high cost lie across the path. This is due to the fact that the latter method does not hold any knowledge about the existence and extent of a polygon, till the graph enters the polygon.

Even after the path enters the polygon the extent of the polygon is still not known and the graph generations keeps wandering in the interior of the polygon, resulting in wasteful graph generation, backtracking and sub-optimal results. The results obtained for some trial examples (see table 4.1 for results) confirm the observations.

The complexity of the method has not been determined formally; is expected to be within O $(n \log n)$ time complexity.

Overheads for sectionalizing search in the *staged search* method, are minimal as the number of sections are of the order of number of polygons in the list. Moreover these

overheads are offset by the reduction in complexity for each section, due to the *guided* nature of search.

5.2 Suggestions for Future Work

Applying *path planning* methods for *path planning* over a *natural terrain* is a relatively unexplored area of work. Heuristic methods developed for such a problem have applications, ranging from a tool for preparation of digitized road-maps for move of military convoys, to a system generating road construction designs in inhospitable terrain.

The broad spectrum of potential applications provide an unlimited scope for research in this area. Some suggestions for such research are listed below:

- To carry out a formalized analysis in terms of complexity, efficiency and optimality of the work proposed here.

- Studying the effects of variation in various parameters such as the STEP size, inter-successor angle, minimum number of successors to be generated etc, and developing heuristics to assign an optimal quantity for these parameters, based on the problem to be solved.

- Two methods of dealing with high cost thematic regions were proposed and illustrated (in figures 4.1 and 4.2 respectively) . One method leaves the generation of a path around such polygons to the successor generation heuristics (in 3.3.2) (*the brute force method*) while the other method sectionalizes the solution at the vertices of the regions likely to lie across the path, thus guiding the search process. Depending on the problem, one of the two methods generate an optimal path. Integrating the two methods in a single set of heuristics, is likely to ensure generation of an optimal path for all problems. One way of doing this is to first generate a path using the "Staged Search" method, include the nodes visited in the global OPEN list and then allow these to compete with those being generated by the default, brute force method.

- Development of additional heuristics for inclusion of roads, rivers and other linear obstacles, to the method proposed. In Dealing with rivers with well known crossing points, a method similar to the *staged search* can be used where the crossing points take over as temporary bounds. For rivers without crossing points and roads, heuristics

proposed by Rowe [Rowe 90] may be used to obtain the crossing, entering and leaving points, and again the *staged search* can be employed.

- Solving the concealment problem (or a visibility problem—just reverse of concealment problem) by a similar method. Once areas not concealed from enemy observation are known, these areas can be declared as *forbidden regions* and the *staged search* can be run then. The problem of finding unconcealed areas is of course to be solved using different methods.

- Solving problems in communication planning, such as designing an optimal relay chain of microwave/ VHF stations between two terminal points. The constraint set for this problem include the constraint on operating range of a VHF(or micro) link. This constraint can be used as the standard STEP or arc length for the problem as a VHF transmission is *line of sight* transmission. Using this step size, the problem reduces to a graph generation, which can be done in a similar manner as proposed in this paper. The cost evaluation function will have different weighted parameters as defined by the equation of *Electromagnetic Energy* received at a receiver station.

Bibliography

[Bagchi 91] Anupam Bagchi.*Application of Fuzzy Logic For Collision Avoidance in Navigation and Manipulation in Dynamic Environments.* Phd Thesis, IIT Kanpur, November 1991.

[Charniac 85] Eugene Charniac and Drew McDermott. *Introduction to Artificial Intelligence.* Addison-wesley. 1985.

[Dong 90] J. Dong, Q. Peng and Y. Liang. *A Stochastic Functional Approach and Terrain Modeling.* Eurographics' 90. Elsevier Science Publishers, B. V. (North Holland).

[Fournier 82] Alain Fournier and Don Russel. *Computer Rendering of Stochastic Models.* Communications of the ACM, Vol 25, No. 6, June 1982, pp 372–384.

[Fowler 79] R. J. Fowler and J. J. little. *Automatic Extraction of Irregular Network Digital Terrain Models.* ACM Trans on Computer graphics, April 1979, pp 199–207.

[Guptill 79] S. C. Guptill. *The Development and Use of Digital Cartographic Databases.* Lecture notes in Computer Science Ser. 81(Database Techniques for Pictorial Applications),Springer Verlag 1979, pp 65–77.

[Hart 68] Peter. E. Hart, N. J. Nilsson and B. Raphael. *A Formal Basis for Heuristic Determination of Minimum Cost Path.* IEEE Trans Systems, man and cybernatics, vol SSC-4,No. 2,July 1968, pp 100–109.

[Jimenez 79] J. Jeminez and J.L Navalon. *The Structure of Queries on Geometric Data.* Lecture notes in Computer Science Ser. 81(Database Techniques for Pictorial Applications),Springer Verlag 1979, pp 219–232.

[Juan 79] Juan Fco. C. Burgueno. *A Geographical Database.* Lecture notes in Computer Science Ser. 81(Database Techniques for Pictorial Applications),Springer Verlag 1979, pp 349–411.

[Khatib 86] K. Khatib. *Real Time Obstacle Avoidance for Manipulators and Mobile Robots.*Int. Journal of Robotics Research, Vol 5, No. 1, spring 1986, pp 90–98.

[Lozano 79] T. Lozano-Peres and M. A. Wesley. *An algorithm for planning Collision Free Paths Among Polyhedral Obstacles.* Comm. ACM vol 22, No. 10, 1979, pp 560-570.

[Mandelbrot 82] B. B. Mandelbrot. *The Fractal Geometry of Nature.* W. H. Freeman an Co. . 1982.

[Mantey 79] P. E. Mantey and E. D. Carlson. *Integrated Geographic Databases : The GADS experience.* Lecture notes in Computer Science Ser. 81(Database Techniques for Pictorial Applications),Springer Verlag 1979, pp 173–198.

[Marshall 80] Robert Marshall and Rodger Wilson. *Procedure models for Generating Three Dimensional Terrain.* Computer Graphics, Vol 14, No 4, 1980, pp 154–161.

[Mitchel 88] J. S. B. Mitchel. *An Alogorithmic Approach to some Problems in Terrain Navigation.* Artificial Intelligence, No.37, 1988, pp 171–201.

[Nilsson 85] N. J. Nilsson *Problem Solving Methods in Artificial Intelligence.* McGraw-Hill,1985.

[Phillips 79] R. L. Phillips. *Definition and Manipulation of graphical Entities in Geographic Information systems.* Lecture notes in Computer Science Ser. 81(Database Techniques for Pictorial Applications), Springer-Verlag 1979, pp 116–133.

[Puttre 92] Michael Puttre. *Geographic Information System Add a New Dimension to Environmental Mapping.* Mechanical Emgineering, June 1992, pp 54–59.

[Rich 83] Elaine Rich. *Artificial Intelligence.* McGraw-Hill, 1983.

[Richbourg 90] N. C. Rowe and R. F. Richbourg. *an efficient Snell's Law method for Optimal Path Planning across multiple,Two-dimensional, Irregular, Homo-*

geneous Cost Regions. International Journal of Robotic Research, Vol 9, No.6, Dec 1990, pp 48–66.

[Rogers 85] David. F. Rogers *Procedural Elements of Computer Graphics.* McGraw-Hill, 1985.

[Rowe 90] Neil. C. Rowe *Roads Rivers and Obstacles : Optimal two dimensional Path planning around linear features for a mobile agent.* International Journal of Robotics Research, volume 9, No. 6, December 1990, pp 67–74.

[Saupe 88] D. Saupe. *Algorithms for Random fractals.* The Science of Fractal Images, Eds: H. O. Peitgen and D. Saupe, Springer Verlag 1988, pp 71–136.

[Starbase] *Starbase Techniques and Tutorials.* Vol 1, 2, 3, Hewlett Packard.